EE 313 Linear Signals & Systems

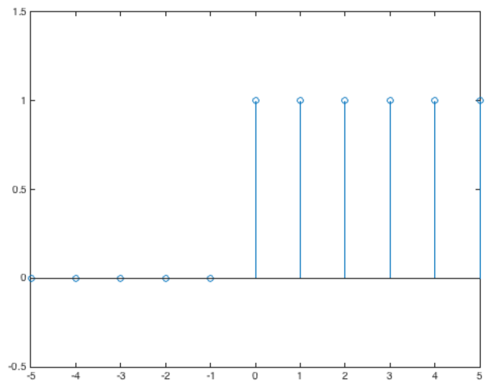
Solution Set for Homework #4 on FIR Filters

By: Anyesha Ghosh & Prof. Brian L. Evans

1. **Prologue:** This problem introduces the convolution sum, and asks you to calculate it for a simple finite impulse response filter (*L*-point averaging filter) given an infinitely long input signal (unit step). The unit step signal models a physical action such as turning on a switch and leaving it on indefinitely. In discrete time, the unit step function $u[n]$ is zero in amplitude for $n < 0$, and one in amplitude for $n \geq 0$.

Solution: (a) The MATLAB function `stepfun(n, n0)` implements $u[n-n_0]$ and is plotted on the right:

```
n = -5:5;
u = stepfun(n, 0);
stem(n, u);
ylim( [-0.5 1.5] );
```



Another way to have computed $u[n]$ in MATLAB is

```
u = ( n >= 0 );
```

For a comparison operation such as >=, MATLAB returns 1 if true and 0 if false.

b) $y[n]$ is calculated as shown below:

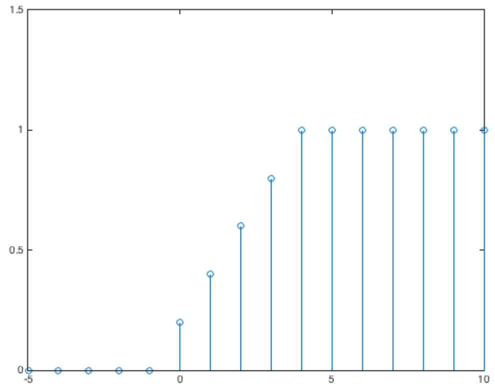
n	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
x[n]	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
x[n-1]	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
x[n-2]	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
x[n-3]	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x[n-4]	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
y[n]	0	0	0	0	0	1/5	2/5	3/5	4/5	1	1	1	1	1	1	1

c) MATLAB code for computing the convolution using the above approach and plotting the result is

```
n = -5:10;
L = 5;
x = stepfun(n, 0);
y = zeros(1, length(n));
for i = 0:L-1
    y = y + x;
    x = [0 x(1:length(n)-1)];        % shifts x right by one sample.
end
y = y / L;
stem(n, y);
ylim( [0 1.5] );
```

A more compact MATLAB program to perform and plot the convolution follows:

```
n = -5:10;
L = 5;
x = stepfun(n, 0);
h = ones(1, L) / L;
y = conv(h, x);
stem(n, y(1:length(n)));
ylim( [0 1.5] );
```



d)

$$y[n] = \frac{1}{L}\sum_{0}^{L-1} x[n-k]\ , x[n] = u[n]$$

$$u[n-k] = \begin{cases} 0, & n < k \\ 1, & n \geq k \end{cases}$$

$$So, y[n] = \begin{cases} 0, & n < 0 \\ \dfrac{n+1}{L}, & 0 \leq n \leq L \ \ (\text{n + 1 of the unit step fns. being added have value 1}) \\ 1, & n > L \end{cases}$$

Epilogue: The averaging filter is a lowpass filter (i.e. low frequencies pass through and high frequencies get attenuated). Can you think why it would be lowpass? This running average is widely used in practice, both on its own, and as a component of other algorithms. It is important enough that there are several hardware designs & algorithms intended to optimize the running average calculation.

2. **Prologue:** Properties provide a way to characterize systems. Using mathematical analyses, one can try different input signals and analyse the responses (outputs). This problem ask you to determine three important system properties— linearity, time-invariance and causality— for several systems by considering signals defined over $-\infty < n < \infty$. The solution set will also point out any differences in the analysis when one can only observe the system for $n \geq 0$, which was not asked in the problem.

Solution:

a) The input-output relationship in the discrete-time domain is $y[n] \ = \ x[n]cos(0.2\pi n)$.
To check time-invariance, input
$x_{shifted}[n] = x[n - n_0]$
and examine the output
$y_{shifted}[n] = \ x[n - n_0]\cos(0.2\ \pi\ n) \ \neq y[n - n_0]$

So, the system is not time-invariant.

To check linearity, consider $y_1[n] = \ x_1[n]\cos(0.2\pi n)\ and\ y_2[n] = \ x_2[n]\cos(0.2\pi n)$

Let $x_{linear}[n] = ax_1[n] + bx_2[n]$

$y_{linear}[n] = x_{linear}[n]\cos(0.2\ \pi\ n) = (ax_1[n] + bx_2[n])\cos(0.2\ \pi\ n)$

$\quad = ax_1[n]cos(0.2\pi n) + bx_2[n]\cos(0.2\pi n) = ay_1[n] + by_2[n]$

So, the system is linear.

Since, *y*[*n*] only depends on *x*[*n*] (the current value), the system is causal.

For observing the system for *n* ≥ 0, there are no initial conditions, and the system is at rest at *n* = 0. We can verify the claim that there are no initial conditions by computing the first few output values: y[0] = x[0] and y[1] = x[1] cos(0.2π) etc. There is no change to the above analyses for part (a).

b) The input-output relationship in the discrete-time domain for the first-order difference FIR filter is $y[n] = x[n] - x[n-1]$.

$Let\ x_{shifted}[n] = x[n-n_0]$

$y_{shifted}[n] = x_{shifted}[n] - x_{shifted}[n-1] = x[n-n_0] - x[n-n_0-1] = y[n-n_0]$

Yes, the system is time invariant.

To check linearity, consider $y_1[n] = x_1[n] - x_1[n-1]$ and $y_2[n] = x_2[n] - x_2[n-1]$

Let $x_{linear}[n] = ax_1[n] + bx_2[n]$

$$y_{linear}[n] = x_{linear}[n] - x_{linear}[n-1] = a\,x_1[n] + b\,x_2[n] - a\,x_1[n-1] - b\,x_2[n-1]$$
$$= a(x_1[n] - x_1[n-1]) + b(x_2[n] - x_2[n-1]) = a\,y_1[n] + b\,y_2[n]$$

So, the system is linear.

Since, *y*[*n*] only depends on x[n] & x[n-1] (current & past value), the system is causal.

For observing the system for *n* ≥ 0, one can see the initial conditions in the system by computing several output values: *y*[0] = *x*[0] – *x*[-1] and *y*[1] = *x*[1] – *x*[0] etc. The FIR filter stores the previous input value x[n-1] in a memory location. At *n* = 0, however, we cannot observe *x*[-1]. Instead, *x*[-1] represents the initial value in the memory location that stores x[*n*-1]. The initial condition must be 0 for the system to be LTI.

c) To check time invariance, consider $y[n] = |x[n]|$.

$Let\ x_{shifted}[n] = x[n-n_0]$

$y_{shifted}[n] = |x_{shifted}[n]| = |x[n-n_0]| = y[n-n_0]$

So, the system is time invariant.

To check linearity, consider $x_1[n] = -x[n]$

$y_1[n] = |x_1[n]| = |x[n]| = y[n] \neq -y[n]$

So, the system is not linear. (Homogeneity part of linearity test was not satisfied)

Since, *y*[*n*] only depends on *x*[*n*] (the current value), the system is causal.

For observing the system for *n* ≥ 0, there is no change to the analysis for LTI properties. The system is a pointwise system; i.e., the output *y*[*n*] only depends on the present input *x*[*n*] and no other quantities that depend on *n*. There are no initial conditions, and hence the system is at rest at *n* = 0.

d) To check time invariance, consider $y[n] = A\,x[n] + B$.

$Let\ x_{shifted}[n] = x[n-n_0]$

$y_{shifted}[n] = A\,x_{shifted}[n] + B = A\,x[n-n_0] + B = y[n-n_0]$

So, the system is time invariant.

To check linearity, consider $y_1[n] = A\, x_1[n] + B \; and \; y_2[n] = A\, x_2[n] + B$

Let $x_{linear}[n] = ax_1[n] + bx_2[n]$

$$y_{linear}[n] = A(a\, x_1[n] + b\, x_2[n]) + B = a(A\, x_1[n] + B) + b(A\, x_2[n] + B) + B(1 - a - b)$$
$$= a\, y_1[n] + \; b\, y_2[n] + B(1 - a - b) \; \neq \; a\, y_1[n] + \; b\, y_2[n] \; (in\ general)$$

So, the system is not linear (in general). It is linear only when B = 0.

Since, *y*[*n*] only depends on *x*[*n*] (the current input value), the system is causal.

For observing the system for *n* ≥ 0, the system must be "at rest" as a necessary condition for LTI properties. That means that *B* = 0.

Epilogue: System properties can be tested through mathematical analysis (as above) by inputting signals and analyzing the outputs (responses). A similar approach can be applied in a lab setting in which one inputs physical signals into a system under test and measures the response.

If y can be represented as a linear function of x, is the system always linear? In this question, we only looked at causal systems. Can you give a simple example of a non-causal system?

3. **Prologue:** We revisit convolution. In this case, you are given the impulse response *h*[*n*] and output *y*[*n*] of the system, and asked to determine the input signal *x*[*n*]. In a camera, for example, light passes through a lens and onto an array of photoreceptors. Intensity of light falling on the photoreceptors is the output *y*[*n*] and an LTI model of the lens gives the impulse response *h*[*n*] which is also known as a point spread function. A goal in camera processing is to find the image *x*[*n*] through deconvolution.

General Solution for Deconvolution: For the case that *x*[*n*], *h*[*n*] and *y*[*n*] are causal, we can develop a systematic approach for solving deconvolution problems. We start by writing out *y*[*n*] = *x*[*n*] * *h*[*n*]:

$$y[n] = h[0]x[n] + h[1]x[n-1] + h[2]x[n-2] + \cdots + h[M]x[n-M]$$

Since *x*[*n*] is causal, the values of *x*[-1], *x*[-2], …, *x*[-*M*] are zero. So, the first output value is

$$y[0] = h[0]x[0]$$

Since we know *h*[*n*] and *y*[*n*], we can solve for x[0] = y[0] / h[0] provided that *h*[0] ≠ 0. Continuing,

$$y[1] = h[0]x[1] + h[1]x[0]$$

This gives one equation in one unknown, *x*[1]. We can continue in this way until all of the values of *x*[*n*] are solved. This approach, when valid, can work for infinitely long causal *x*[*n*] signals. The approach is valid none of the values of *h*[*n*] over its extent is zero. See the epilogue for a more general method.

Solution:

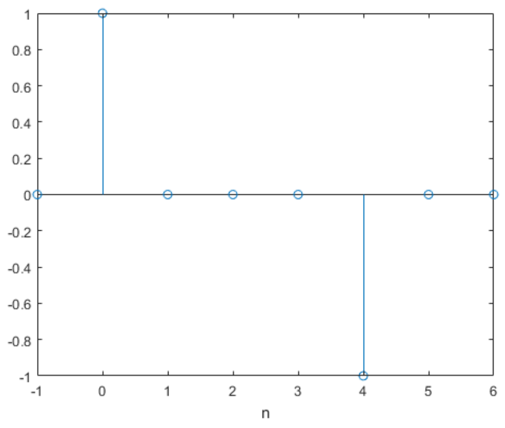
a) $h[n] = \delta[n-2]$ and $y[n] = u[n-3] - u[n-6]$. Here, *y*[*n*] is a rectangular pulse with amplitude 1 for discrete-time samples *n* = { 3, 4, 5 }.
$y[n] = h[n] * x[n] = h[0]x[n] + h[1]x[n-1] + h[2]x[n-2] + \cdots + h[M]x[n-M] = x[n-2]$
That means $x[n-2] = u[n-3] - u[n-6]$

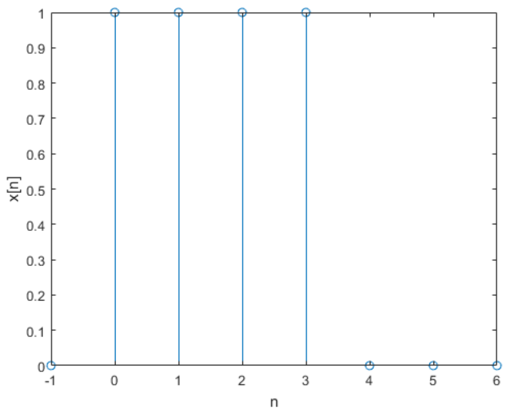
$\rightarrow \; x[n] = u[n-1] - u[n-4]$

b) $h[n] = \; \delta[n] - \delta[n-1]$ and $y[n] = \delta[n] - \delta[n-4]$.
We can use the general solution above.

$y[0] = h[0]x[0] = 1$ which means $x[0] = y[0]/h[0] = 1$
$y[1] = h[0]x[1] + h[1]x[0] = x[1] - 1 = 0$ which means $x[1] = 1$
$y[2] = h[0]x[2] + h[1]x[1] = x[2] - 1 = 0$ which means that $x[2] = 1$
$y[3] = h[0]x[3] + h[1]x[2] = x[3] - 1 = 0$ which means that $x[3] = 1$
$y[4] = h[0]x[4] + h[1]x[3] = x[4] - 1 = -1$ which means that $x[4] = 0$
$y[5] = h[0]x[5] + h[1]x[4] = x[5] - 0 = 0$ which means that $x[5] = 0$

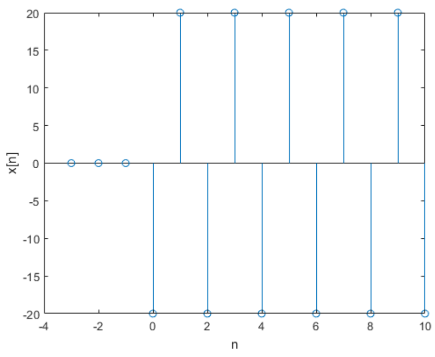
So, $x[n] = u[n] - u[n-4]$. Below, $y[n]$ is plotted on the left, and $x[n]$ on the right.

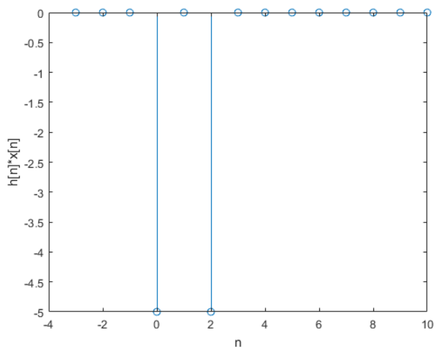




c) Impulse response of a four-point averaging filter is
$h[n] = \frac{1}{4}(\delta[n] + \delta[n-1] + \delta[n-2] + \delta[n-3])$
and the output is $y[n] = -5\delta[n] - 5\delta[n-2]$.
We can use the general solution above.
$y[0] = h[0]x[0] = -5$ which means $x[0] = \frac{y[0]}{h[0]} = -20$
$y[1] = h[0]x[1] + h[1]x[0] = \frac{1}{4}x[1] - 5 = 0$ which means $x[1] = 20$
$y[2] = h[0]x[2] + h[1]x[1] + h[2]x[0] = \frac{1}{4}x[2] + 5 - 5 = -5$ which means that $x[2] = -20$
$y[3] = h[0]x[3] + h[1]x[2] + h[2]x[1] + h[3]x[0] = \frac{1}{4}x[3] - 5 + 5 - 5 = 0$ which is $x[3] = 20$
$y[3] = h[0]x[4] + h[1]x[3] + h[2]x[2] + h[3]x[1] = \frac{1}{4}x[4] + 5 - 5 + 5 = 0$ which is $x[3] = -20$
and the pattern continues indefinitely.

So, $x[n] = -20\ (-1)^n\ u[n]$. Below, x[n] is plotted on the left, and $y[n]$ on the right.





Alternate solution: $x[n]$ can also be calculated using some basic algebra:

n	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**…**
y[n]	-5	0	-5	0	0	0	0	0	0	…
x[n]	a	b	c	d	e	f	g	h	i	…
x[n-1]	0	a	b	c	d	e	f	g	h	…
x[n-2]	0	0	a	b	c	d	e	f	g	…
x[n-3]	0	0	0	a	b	c	d	e	f	…

Solving the equations from left to right, we get,
a/4 = -5 => a = -20
a+b = 0 => b = -a = 20
(a+b+c)/4 = -5 => c = -20
a+b+c+d = 0 => d = 20,….and so on

So, we get

n	0	1	2	3	4	5	6	7	8	…
x[n]	-20	20	-20	20	-20	20	-20	20	-20	…

Which is exactly what we had seen above.

Epilogue: When convolving two ***finite*** discrete-time signals $h[n]$ of length N_h samples and $x[n]$ of N_x samples, the result $y[n] = x[n] * h[n]$ will be $N_y = N_h + N_x - 1$ samples in duration. We can apply the same relationship in deconvolution to solve for N_x. In (a), we have $N_x = 3 - 1 + 1 = 3$ samples. In (b), $N_x = 5 - 2 + 1 = 4$. In (c), $N_x = 3 - 4 + 1 = 0$. Because $N_x = 0$, it will take an infinitely long $x[n]$ in (c).

When any of the impulse response coefficients is zero, we can take the same approach as above to develop a system of equations in the form **H x** = **y** where **H** is an N x N matrix, **x** is a vector of the N unknown values of $x[n]$ and **y** is a vector of $y[n]$ values, and solve for **x**. This approach only works for a finite number of unknown $x[n]$ values. The matrix **H** is known as the convolution matrix:

Let's consider a 3 x 3 example:

$$y[0] = h[0]x[0]$$

$y[1] = h[1]x[0] + h[0]x[1]$
$y[2] = h[2]x[0] + h[1]x[1] + h[0]x[2]$

$$\begin{bmatrix} h[0] & 0 & 0 \\ h[1] & h[0] & 0 \\ h[2] & h[1] & h[0] \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \end{bmatrix} = \begin{bmatrix} y[0] \\ y[1] \\ y[2] \end{bmatrix}$$

And then we seek to solve for the **x** vector.

4. **Prologue:** The questions concerns associativity and commutativity properties of the convolution sum. This question introduces a cascade, which is used to build complex blocks from simpler ones.

Solution:
a) For any system, $h[n] = y[n]$ when $x[n] = \delta[n]$
So, $h_1[n] = \delta[n] - \delta[n-1], h_2[n] = \delta[n] + \delta[n-2], h_3[n] = \delta[n-1] + \delta[n-2]$

b) $h[n] = h_1[n] * h_2[n] * h_3[n]$
First calculate $h_1[n] * h_2[n]$.

$$\begin{aligned} h_1[n] * h_2[n] &= \sum h_1[k]h_2[n-k] = \sum (\delta[k] - \delta[k-1])(\delta[n-k] + \delta[n-k-2]) \\ &= \sum \delta[k](\delta[n-k] + \delta[n-k-2]) - \sum \delta[k-1](\delta[n-k] + \delta[n-k-2]) \\ &= \delta[n] + \delta[n-2] - \delta[n-1] + \delta[n-3] \end{aligned}$$

$$\begin{aligned} h[n] &= (h_1[n] * h_2[n]) * h_3[n] \\ &= \sum (\delta[k] - \delta[k-1] + \delta[k-2] - \delta[k-3])(\delta[n-k-1] + \delta[n-k-2]) \\ &= \sum (\delta[k])(\delta[n-k-1] + \delta[n-k-2]) - \sum (\delta[k-1])(\delta[n-k-1] + \delta[n-k-2]) \\ &\qquad + \sum (\delta[k-2])(\delta[n-k-1] + \delta[n-k-2]) - \sum (\delta[k-3])(\delta[n-k-1] + \delta[n \\ &\qquad - k-2]) \\ &= \delta[n-1] + \delta[n-2] - \delta[n-2] - \delta[n-3] + \delta[n-3] + \delta[n-4] - \delta[n-4] - \delta[n-5] \\ &= \delta[n-1] - \delta[n-5]. \end{aligned}$$

c) $h[n] = \delta[n-1] - \delta[n-5]$
Since, $h[n] = y[n]\ when\ x[n] = \delta[n]$, we can read off the difference equation from the expression for h[n].

d) The difference equation is: $y[n] = x[n-1] - x[n-5]$

Epilogue: Given what we know about the convolution sum properties, do you think that re-ordering the filter blocks would change the output? Would your answer change if we fed back the final output to the first block in the cascade? What if we fed it back to one of the intermediate blocks?